Lane Detection

```
TEAM MEMBERS:
1.Ujwal Kothapally
2.Venkata Krishna Sreekar Padakandla
```

Install Roboflow Package

In [3]:
```
!pip install roboflow
```

```
Requirement already satisfied: roboflow in /usr/local/lib/python3.10/dist-pac
kages (1.1.9)
Requirement already satisfied: certifi==2023.7.22 in /usr/local/lib/python3.1
0/dist-packages (from roboflow) (2023.7.22)
Requirement already satisfied: chardet==4.0.0 in /usr/local/lib/python3.10/di
st-packages (from roboflow) (4.0.0)
Requirement already satisfied: cycler==0.10.0 in /usr/local/lib/python3.10/di
st-packages (from roboflow) (0.10.0)
Requirement already satisfied: idna==2.10 in /usr/local/lib/python3.10/dist-p
ackages (from roboflow) (2.10)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.1
0/dist-packages (from roboflow) (1.4.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-p
ackages (from roboflow) (3.7.1)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dis
t-packages (from roboflow) (1.23.5)
Requirement already satisfied: opencv-python-headless==4.8.0.74 in /usr/loca
l/lib/python3.10/dist-packages (from roboflow) (4.8.0.74)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dis
t-packages (from roboflow) (9.4.0)
Requirement already satisfied: pyparsing==2.4.7 in /usr/local/lib/python3.10/
dist-packages (from roboflow) (2.4.7)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/d
ist-packages (from roboflow) (2.8.2)
Requirement already satisfied: python-dotenv in /usr/local/lib/python3.10/dis
t-packages (from roboflow) (1.0.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-pac
kages (from roboflow) (2.31.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from roboflow) (1.16.0)
Requirement already satisfied: supervision in /usr/local/lib/python3.10/dist-
packages (from roboflow) (0.16.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.10/d
ist-packages (from roboflow) (2.0.7)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.10/dist
-packages (from roboflow) (4.66.1)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dis
t-packages (from roboflow) (6.0.1)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.1
0/dist-packages (from roboflow) (1.0.0)
Requirement already satisfied: python-magic in /usr/local/lib/python3.10/dist
-packages (from roboflow) (0.4.27)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/
dist-packages (from matplotlib->roboflow) (1.2.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib->roboflow) (4.44.3)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib->roboflow) (23.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
hon3.10/dist-packages (from requests->roboflow) (3.3.2)
Requirement already satisfied: scipy<2.0.0,>=1.9.0 in /usr/local/lib/python3.
10/dist-packages (from supervision->roboflow) (1.11.3)
```

Import Dependencies

In [4]:
```
from IPython.display import Image, clear_output
```

Install PyTorch and torchvision

In [2]:
```
!pip install torch
!pip install torchvision
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packag
es (2.1.0+cu118)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pac
kages (from torch) (3.13.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.1
0/dist-packages (from torch) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packag
es (from torch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-pac
kages (from torch) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packa
ges (from torch) (3.1.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packa
ges (from torch) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dis
t-packages (from torch) (2.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/d
ist-packages (from jinja2->torch) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist
-packages (from sympy->torch) (1.3.0)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-
packages (0.16.0+cu118)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packag
es (from torchvision) (1.23.5)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-pac
kages (from torchvision) (2.31.0)
Requirement already satisfied: torch==2.1.0 in /usr/local/lib/python3.10/dist
-packages (from torchvision) (2.1.0+cu118)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python
3.10/dist-packages (from torchvision) (9.4.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pac
kages (from torch==2.1.0->torchvision) (3.13.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.1
0/dist-packages (from torch==2.1.0->torchvision) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packag
es (from torch==2.1.0->torchvision) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-pac
kages (from torch==2.1.0->torchvision) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packa
ges (from torch==2.1.0->torchvision) (3.1.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packa
ges (from torch==2.1.0->torchvision) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dis
t-packages (from torch==2.1.0->torchvision) (2.1.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
hon3.10/dist-packages (from requests->torchvision) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist
-packages (from requests->torchvision) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
0/dist-packages (from requests->torchvision) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
0/dist-packages (from requests->torchvision) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/d
ist-packages (from jinja2->torch==2.1.0->torchvision) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist
-packages (from sympy->torch==2.1.0->torchvision) (1.3.0)
```

In [5]:
```
!pip install --upgrade pip
```

```
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-packages
(23.1.2)
Collecting pip
  Downloading pip-23.3.1-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 12.5 MB/s eta 0:00:0
0
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.1.2
    Uninstalling pip-23.1.2:
      Successfully uninstalled pip-23.1.2
Successfully installed pip-23.3.1
```

Import PyTorch and Other Libraries

In [6]:
```
from roboflow import Roboflow
import torch
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
from torchvision.datasets import ImageFolder
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import torch.nn as nn
import torchvision.models as models
from torchvision import transforms
import torch.optim as optim
```

Downloads a dataset from Roboflow using the YOLOv5 model format, specifying the API key, project name and version, and saves it

In [7]:
```
rf = Roboflow(api_key="8XxgFmL7Zjg4BCle3kc4", model_format="yolov5")
dataset = rf.workspace().project("lane_detection-rd6mu").version(1).download(l
```

```
loading Roboflow workspace...
loading Roboflow project...

Downloading Dataset Version Zip in /Downloads to yolov5pytorch:: 100%|
| 13630/13630 [00:00<00:00, 38871.19it/s]



Extracting Dataset Version Zip to /Downloads in yolov5pytorch:: 100%|
| 320/320 [00:00<00:00, 6476.31it/s]
```

Custom PyTorch dataset class (CustomDataset) that wraps around the ImageFolder class, allowing for **customization of the data loading process, sets a seed for reproducibility, defines image transformations

In [8]:
```
class CustomDataset(Dataset):
    def __init__(self, root_dir, transform=None):
        self.dataset = ImageFolder(root_dir, transform=transform)

    def __len__(self):
        return len(self.dataset)

    def __getitem__(self, idx):
        return self.dataset[idx]

# Set seed for reproducibility
torch.manual_seed(42)

# Define transforms
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
])

# Load dataset
dataset = CustomDataset(root_dir="/Downloads", transform=transform)
```

Custom PyTorch neural network model (LaneDetectionModel) based on the DeepLabV3 architecture with a ResNet-101 backbone, loads pre-trained weights, and modifies the output layer to suit the task of lane detection by changing the number of output classes.

In [9]:
```
class LaneDetectionModel(nn.Module):
    def __init__(self, num_classes=1):
        super(LaneDetectionModel, self).__init__()

        # Load pre-trained DeepLabV3 model
        self.deeplabv3 = models.segmentation.deeplabv3_resnet101(pretrained=Tr

        # Modify the output layer based on your needs
        in_channels = self.deeplabv3.classifier[-1].in_channels
        self.deeplabv3.classifier[-1] = nn.Conv2d(in_channels, num_classes, ke

    def forward(self, x):
        return self.deeplabv3(x)['out']

# Create an instance of the model
your_model = LaneDetectionModel()

# Print the model architecture
print(your_model)
```

```
ack_running_stats=True)
          (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bi
as=False)
          (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, t
rack_running_stats=True)
          (relu): ReLU(inplace=True)
        )
        (15): Bottleneck(
          (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bi
as=False)
          (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, tr
ack_running_stats=True)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), pad
ding=(2, 2), dilation=(2, 2), bias=False)
          (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, tr
ack_running_stats=True)
          (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bi
as=False)
          (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, t
rack_running_stats=True)
```

Instantiates an instance of the previously defined LaneDetectionModel with one output class, sets up Mean Squared Error (MSE) loss, defines an Adam optimizer with a learning rate of 0.001, specifies the number of training epochs (5), and splits the dataset into training, validation, and test sets using random splitting. Finally, creates DataLoader instances for the training, validation, and test sets with batch size 8

In [10]:
```
your_model = LaneDetectionModel(num_classes=1)
criterion = nn.MSELoss()
# Define optimizer (e.g., Adam)
optimizer = optim.Adam(your_model.parameters(), lr=0.001)
num_epochs = 5
# Split the dataset into train, validation, and test sets
train_size = int(0.8 * len(dataset))
val_size = int(0.1 * len(dataset))
test_size = len(dataset) - train_size - val_size

train_dataset, temp_dataset = torch.utils.data.random_split(dataset, [train_si
val_dataset, test_dataset = torch.utils.data.random_split(temp_dataset, [val_s

# Create DataLoader
train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=8, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=8, shuffle=False)
```

Set of image transforms, including resizing, random horizontal flipping, color jittering, converting to a PyTorch tensor, and normalization.

In [11]:
```
# Define transforms for normalization and augmentation
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
])

# Load dataset with the defined transforms
dataset = CustomDataset(root_dir="/Downloads", transform=transform)
```

A function (visualize_samples) to visualize the first sample of each minibatch from a given DataLoader, creates a DataLoader for visualization with a batch size of 8 and shuffling

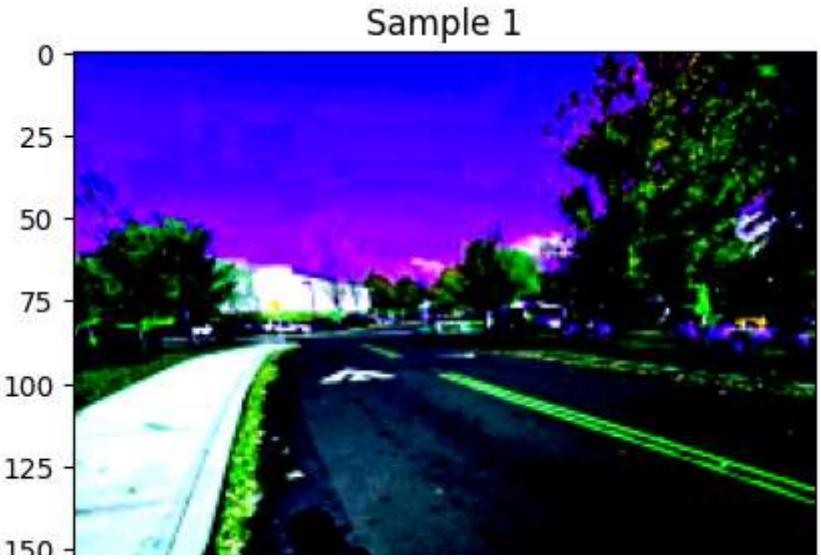
In [12]:
```
# Visualize the 1st sample of each minibatch of size 8
def visualize_samples(loader):
    data_iter = iter(loader)
    images, _ = next(data_iter)

    for i in range(images.shape[0]):
        plt.figure()
        plt.imshow(images[i].permute(1, 2, 0))
        plt.title(f"Sample {i+1}")
        plt.show()

# Create DataLoader for visualization
visualize_loader = DataLoader(dataset, batch_size=8, shuffle=True)

# Visualize the 1st sample of each minibatch
visualize_samples(visualize_loader)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
with RGB data ([0..1] for floats or [0..255] for integers).
```



Set the neural network model (your_model) to training mode, iterates through epochs and mini-batches in the training DataLoader (train_loader), performs a forward pass, calculates the Mean Squared Error (MSE) loss between the model's output and the target, backpropagates the gradients, and updates the model's weights using the Adam optimizer.

In [13]:
```
your_model.train()  # Set the model to training mode
#target = target.view(-1, 1, 1, 1).expand_as(output)
for epoch in range(num_epochs):
    for batch_idx, (data, target) in enumerate(train_loader):
        if batch_idx == 0:  # Only for the first mini-batch
            optimizer.zero_grad()  # Zero the gradients
            output = your_model(data)  # Forward pass
            target = target.view(-1, 1, 1, 1).expand_as(output).float()
            print(f"Model Output Size: {output.shape}")
            loss = criterion(output, target)  # Calculate the loss
            loss.backward()  # Backward pass
            optimizer.step()  # Update the weights
            print(f"Epoch {epoch+1}, Batch {batch_idx+1}, Loss: {loss.item()}"
```

```
Model Output Size: torch.Size([8, 1, 224, 224])
Epoch 1, Batch 1, Loss: 1.4757877588272095
Model Output Size: torch.Size([8, 1, 224, 224])
Epoch 2, Batch 1, Loss: 2.2448413372039795
Model Output Size: torch.Size([8, 1, 224, 224])
Epoch 3, Batch 1, Loss: 1.8363642692565918
Model Output Size: torch.Size([8, 1, 224, 224])
Epoch 4, Batch 1, Loss: 1.3391704559326172
Model Output Size: torch.Size([8, 1, 224, 224])
Epoch 5, Batch 1, Loss: 0.8838760852813721
```